

# Tema 1: Fundamentos de concurrencia

*Desarrollo de Aplicaciones Multiplataforma*

*Programación de Servicios y Procesos*

*Departamento de Informática  
IES nº 11. La Cerámica*

# Objetivos didácticos

- Diferenciar entre **programa y proceso**.
- Comprender qué es la **conurrencia**.
- Conocer el *concepto, diferencias y relación* existente entre las dos unidades básicas de ejecución: **procesos e hilos**
- Tener nociones sobre **programación concurrente**.

# Contenidos

- ❑ **Diferencia entre programa y proceso**
- ❑ **Concurrencia**
  - ✓ Concurrencia software Vs. Paralelismo hardware
  - ✓ ¿Dónde surge la concurrencia?
  - ✓ Concurrencia inherente vs. Concurrencia potencial
- ❑ **Unidades de ejecución: Procesos e hilos**
  - ✓ Concepto de proceso e hilo
  - ✓ Procesos Vs. Hilos
  - ✓ Relación entre procesos e hilos
- ❑ **Programación concurrente**
  - ✓ Concurrencia en un programa
  - ✓ ¿Cómo expresar la concurrencia?
  - ✓ Características de los sistemas concurrentes
  - ✓ Beneficios de la programación concurrente
  - ✓ Dificultades de la programación concurrente
  - ✓ Aplicaciones de la programación concurrente

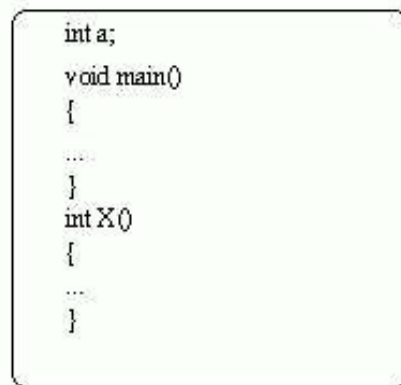
# Diferencia entre programa y proceso (1)

- **Programa**

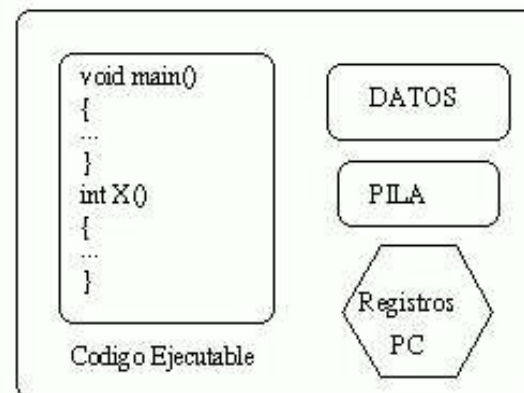
- Se trata de una **entidad pasiva**.
  - Conjunto de **instrucciones u órdenes** que indican a la máquina las operaciones que ésta debe realizar con unos datos determinados, es decir, indican a la computadora como obtener unos datos de salida a partir de unos datos de entrada.

- **Proceso**

- Se trata de una **entidad activa**.
  - Un proceso es un **programa en ejecución**

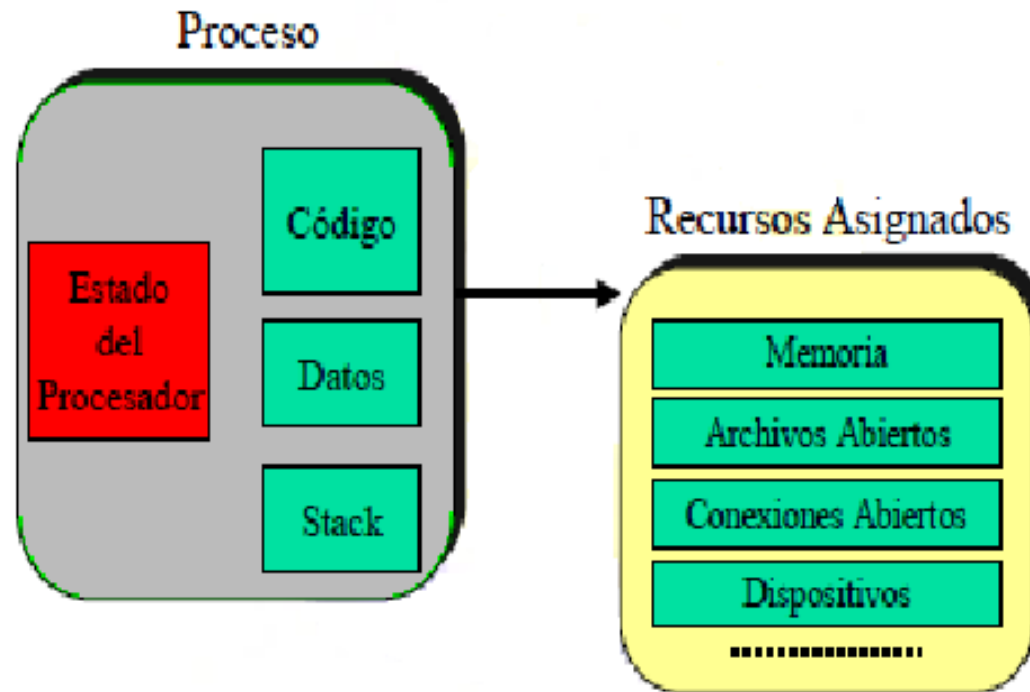


PROGRAMA



PROCESO

# Diferencia entre programa y proceso (2)



- El proceso es una abstracción creada por el sistema operativo, que **se compone de**:
  - **Programa**: *Código* y *datos* del programa cargado en memoria principal.
  - **Contexto de ejecución**: *PC*, *registros de procesador* y una *pila* para invocación de procedimientos.

# Concurrencia (1)

- En el *mundo real*, la concurrencia es la **simultaneidad de hechos**.



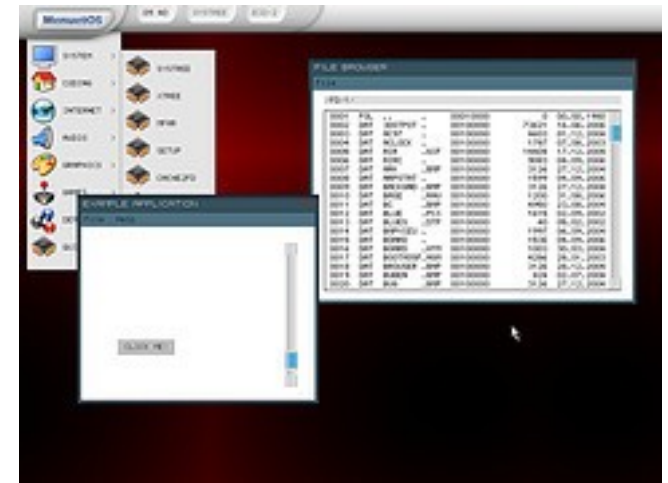
# Concurrencia (2)

- **Ejemplos** de concurrencia en la **vida real**
  - Varias personas participando en una actividad colectiva: fiesta, reunión, viajando juntos.
  - Las actividades de una empresa donde hay varias máquinas trabajando en un mismo período de tiempo.
  - Una persona que está atendiendo a su trabajo y a la vez responde teléfono, consultas por Internet, oye música y toma un café.
- Puede aprovecharse para colaborar en **realizar una tarea compleja, dividiéndola en tareas menores**.
- Se presentan situaciones de **competencia por los recursos**, incluso a nivel de lucha por ellos.
- Surgen problemas de **comunicación y sincronización**.



# Concurrencia (3)

- **Ejemplos** de concurrencia en **computación**
  - Las computadoras personales permiten varios programas ejecutándose simultáneamente:
    - Sistema Operativo,
    - Antivirus,
    - Internet (navegador, varios sitios),
    - Chat,
    - Procesador de textos,
    - juegos.
  - Un mismo programa permite simultáneamente:
    - Varias ventanas activas
    - Varios elementos gráficos activos en una ventana
    - Entrada por teclado
    - Entrada por ratón
    - Entrada por otros dispositivos
    - Uso de impresora





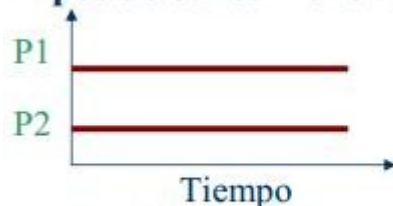
# Concurrencia (4)

- En *informática*, el concepto de concurrencia va ligado indisolublemente al concepto de **proceso** (programa en ejecución).
- Podemos encontrarnos con **dos definiciones**:

## 1. Concurrencia real o paralelismo

Dos procesos, P1 y P2, son concurrentes cuando se ejecutan de manera que sus intervalos se solapan.

**Tipos de concurrencia:**

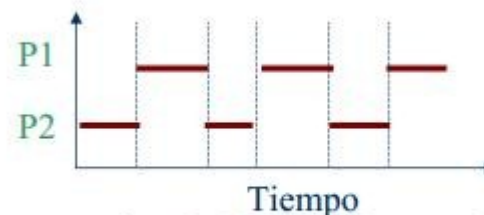


**Real o paralelismo**

Nº procesadores  $\geq$  Nº procesos

## 2. Concurrencia aparente, simulada o pseudoparalelismo

Dos procesos, P1 y P2, se ejecutan concurrentemente si la primera instrucción de P1 se ejecuta entre la primera y la última instrucción de P2.



**Aparente, simulada o pseudoparalelismo**

Nº procesadores  $<$  Nº procesos

# Concurrencia (5)

- Concurrencia real

- Cada proceso se ejecuta en un procesador.

- Se produce una ejecución en paralelo.
    - Paralelismo real.

4 CPUs



- Concurrencia aparente

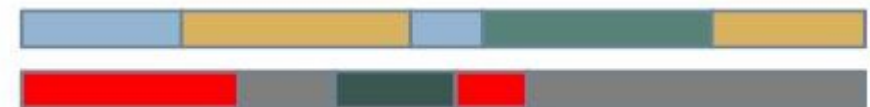
- Hay más procesos que procesadores

- Los procesos se multiplexan en el tiempo.
    - Pseudoparalelismo.


1 CPU



2 CPUs



# Concurrencia (6)

- Ambas definiciones anteriores, implican que, se habla de concurrencia cuando hay una **existencia simultánea de varios procesos en ejecución**.
  -  Existencia simultánea no implica ejecución simultánea.
    - Dependerá del hardware subyacente.
  - El paralelismo es un caso particular de la concurrencia.
  - Se habla de paralelismo hardware cuando ocurre la ejecución simultánea de instrucciones.

# Concurrencia Software Vs. Paralismo Hardware

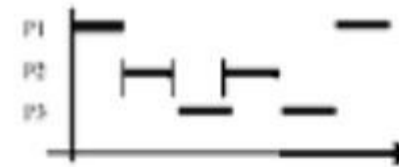
- La concurrencia software es un concepto lógico , no implica la existencia de paralelismo hardware.
  - Las operaciones hardware ocurren en paralelo si ocurren al mismo tiempo.
  - Las operaciones (software) en un programa son concurrentes si pueden ejecutarse en paralelo, aunque no necesariamente deben ejecutarse así.

# ¿Dónde surge la concurrencia?

- En informática la concurrencia surge en los siguientes entornos hardware:
  - Sistemas con **un solo procesador** (sistemas monoprocesador)
  - Sistemas con **más de un procesador** (sistemas multiprocesador)
    - Sistemas estrechamente acoplados (**multiprocesadores**)
    - Sistemas débilmente acoplados (sistemas **distribuidos**)

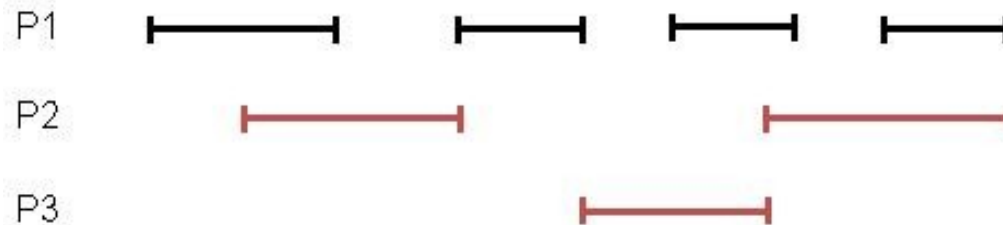
# ¿Dónde surge la concurrencia?. Sistema monoprocesador

- **Sistema multiprogramado** (con un único procesador o monoprocesador)
  - Los procesos multiplexan o **intercalan** sus ejecuciones sobre un único procesador para dar la apariencia de simultaneidad.
  - No hay paralelismo
  - Los procesos comparten la memoria
  - Forma “natural” de sincronizar y comunicar procesos
    - Variables compartidas



# ¿Dónde surge la concurrencia?. Sistema multiprocesador

- **Sistema Multiprocesador** (sistema con más de un procesador)
  - Los procesos se **superponen**
  - Existe paralelismo real entre los procesos

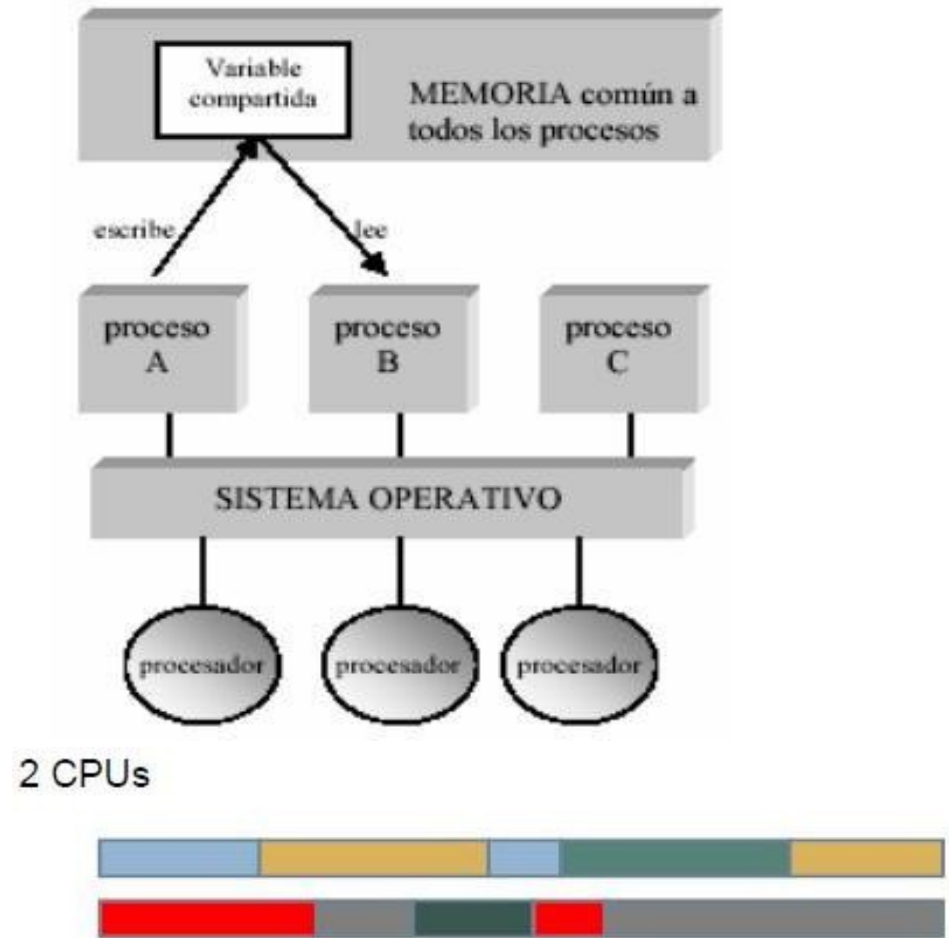




# ¿Dónde surge la concurrencia?.

## Sistema multiprocesador estrechamente acoplado

- **Multiprocesador estrechamente acoplado**
  - Los procesos comparten la memoria
  - La sincronización y comunicación entre procesos se suele hacer mediante variables compartidas.
  - Normalmente se combinan paralelismo real y pseudoparalelismo



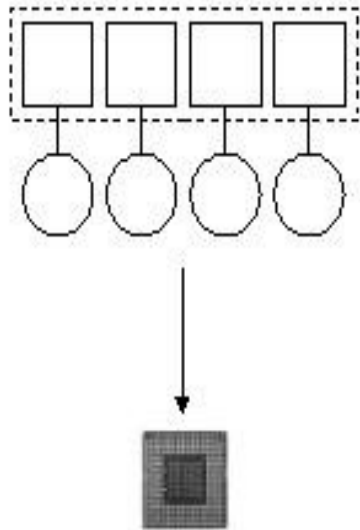
# ¿Dónde surge la concurrencia?.

## Sistema multiprocesador débilmente acoplado o distribuido

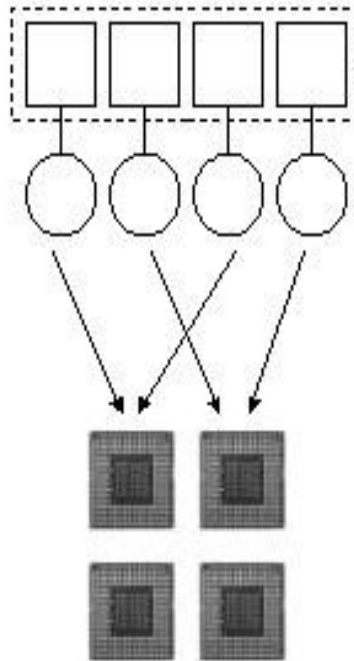
- **Sistema débilmente acoplado o distribuido**
  - Compartición de recursos dispersos
  - La forma natural de comunicar y sincronizar procesos es mediante el uso de paso de mensajes



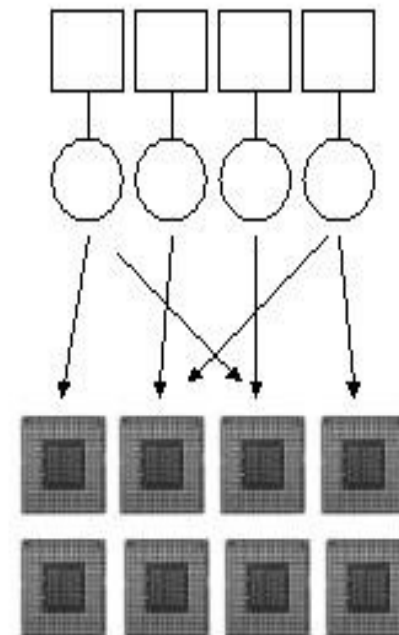
# ¿Dónde surge la concurrencia?. Síntesis



multiprogramación



multiproceso



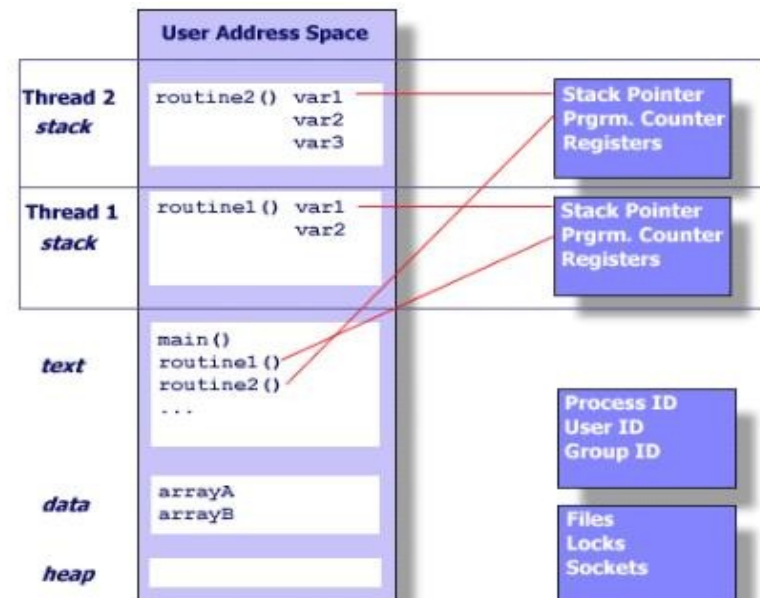
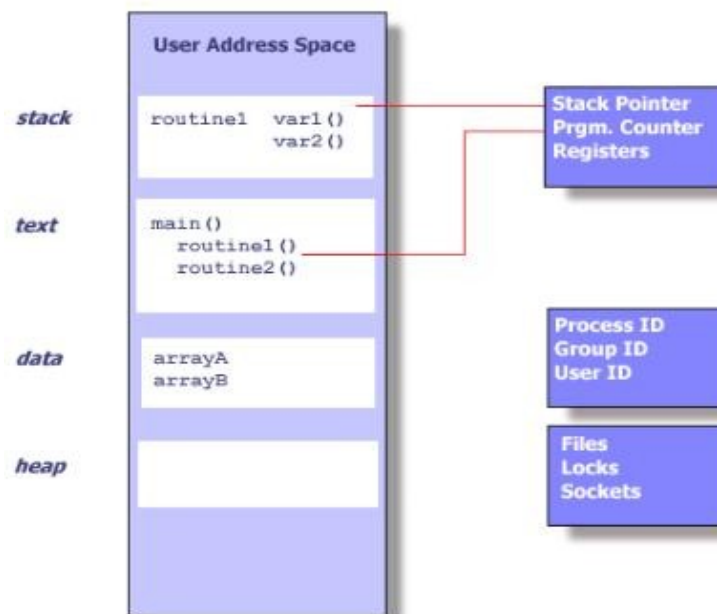
Procesamiento  
distribuido

# Concurrencia inherente Vs. Concurrencia potencial

- **Sistemas inherentemente concurrentes**
  - El entorno con el que interactúan, o el entorno que modelan tiene forzosamente actividades simultáneas.
  - **Por ejemplo:** Red de cajeros automáticos
- **Sistemas potencialmente concurrentes**
  - No es estrictamente necesario que haya concurrencia, pero se puede sacar partido de ella, por ejemplo para aumentar la velocidad.
  - **Por ejemplo:** Un programa que muestre noticias sobre la bolsa (los datos del mercado provienen de un equipo remoto) e información (almacenada en una BD local) sobre la cartera de valores de un usuario en la misma ventana. Como las noticias del mercado no dependen de los datos de la cartera de valores, las dos tareas se pueden ejecutar de forma paralela.

# Unidades de ejecución: concepto de proceso e hilo

- Existen básicamente **dos unidades de ejecución**:
  - Procesos**: En el contexto de un sistema operativo, un proceso es una instancia de un Programa en ejecución. Los procesos pueden ser:
    - Secuenciales: Poseen un único flujo de control (**monohilo**)
    - Concurrentes: Poseen varios hilos de ejecución (**multihilo**).
  - Hilos**: En el contexto de un programa concurrente, un Hilo (Thread) es cada uno de los flujos secuenciales de control independientes especificados en el programa.



# Unidades de ejecución: Procesos Vs. Hilos

- **Procesos**

- **Constan de al menos uno o más hilos de ejecución.**
- **Generalmente son muy independientes** unos de otros, de tal forma que podrían pararse unos sin que se vean afectados los demás.
- **Son gestionados por el propio Sistema operativo.**
  - Es el propio sistema el que se encarga de la **creación y eliminación** de procesos cuidando asimismo de la **independencia entre ellos** (Por ejemplo, cada proceso tiene su propio contador de programa, variables, registros...) precisamente por la seguridad y estabilidad del propio sistema operativo.
  - El sistema se encarga de la **planificación de los procesos** (maximizando la utilización del procesador)
  - En caso de que dichos procesos tuvieran que interactuar lo harían a través de **mecanismos de comunicación dados por el sistema**, por ejemplo a través de interrupciones.

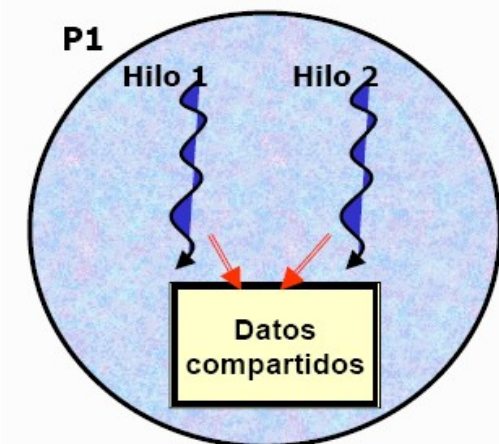
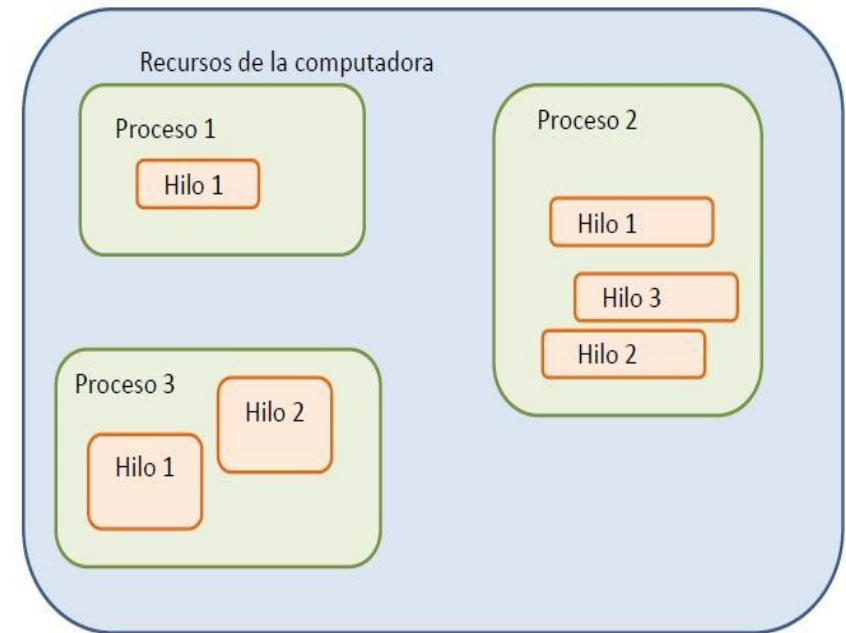
- **Hilos**

- **Un hilo siempre existe con un proceso**
- **Generalmente comparten recursos de forma directa** (proceso ligero, no posee recursos propios sino que los comparte con otros)
- **Son gestionados por la propias aplicaciones en ejecución:**
  - Por ejemplo, una aplicación realizada en el lenguaje Java puede:
    - Generar varios hilos de ejecución.
    - Detenerlos, cuando sea necesario.
    - Sincronizarlos para que no generen conflictos al actuar sobre un mismo objeto, etc.
  - La comunicación también corre a cargo del programa, usando estructuras de datos.



# Unidades de ejecución: Relación entre procesos e hilos

- Un **proceso** consta de uno o más hilos de ejecución
  - **Monohilo**: Existe un único flujo de control o hilo para controlar su ejecución
  - **Multihilo**: Existen varios flujos o hilos de ejecución
- Todos los **hilos generados por un proceso comparten** toda la memoria reservada para el proceso (es decir, su espacio de direcciones que incluye: la sección de código, la sección de datos) y los recursos del sistema operativo, tales como archivos abiertos.
- No obstante, **cada hilo tiene** su propio *contador de programa, pila de ejecución y estado CPU* incluyendo el valor de los registros.





# Programación concurrente

- Disciplina que se encarga del estudio de las notaciones que permiten especificar la ejecución concurrente de las acciones de un programa, así como las técnicas para resolver los problemas inherentes a la ejecución concurrente (comunicación y sincronización).
- Tradicionalmente estuvo asociada al mundo de los Sistemas Operativos.
- **Primeros programas concurrentes: Sistemas Operativos**
  - Evolución plataformas hardware
  - Diferentes partes del sistema operativo en ejecución sin un orden predecible y compartiendo variables → nuevos problemas.
  - Soluciones específicas
- Tres **hitos importantes marcan su evolución**
  - Hilo
  - Aparición de lenguajes de alto nivel que den soporte a la PC
  - La aparición de Internet.

# Programación concurrente.

## Concurrencia en un programa

### PROGRAMA MONOHILO

```
private void ponHora() {
    ahora = Calendar.getInstance();

    // valores numéricos de horas, minutos y segundos
    int hora = ahora.get(Calendar.HOUR_OF_DAY);
    int minuto = ahora.get(Calendar.MINUTE);
    int segundo = ahora.get(Calendar.SECOND);

    // los convertimos a String
    String fSegundo = String.format("%02d", segundo);
    String fMinuto = String.format("%02d", minuto);
    String fHora = String.format("%02d", hora);

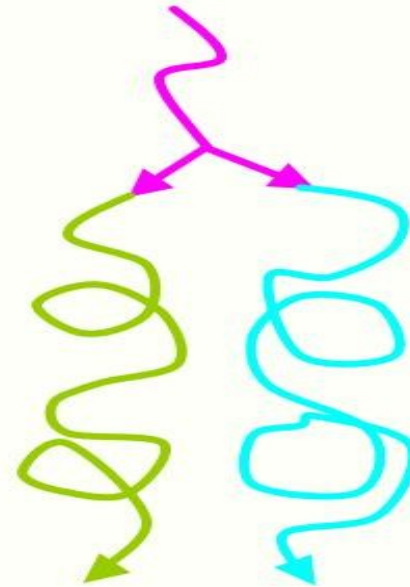
    // añadimos un poco de formato
    String fSegundo = String.format("%02d", segundo);
    String fMinuto = String.format("%02d", minuto);
    String fHora = String.format("%02d", hora);
    String fResultado = (fHora + ":" + fMinuto + ":" + fSegundo);

    // mostramos el resultado
    EditText etHora = findViewById(R.id.etHora);
    EditText etMinuto = findViewById(R.id.etMinuto);
    EditText etSegundo = findViewById(R.id.etSegundo);
    // Pon hora
}

// inicialización de componentes
private void init() {
    // Etiqueta hora
    TextView tvHora = findViewById(R.id.tvHora);
    tvHora.setText(fHora);
    // Etiqueta minuto
    TextView tvMinuto = findViewById(R.id.tvMinuto);
    tvMinuto.setText(fMinuto);
    // Etiqueta segundo
    TextView tvSegundo = findViewById(R.id.tvSegundo);
    tvSegundo.setText(fSegundo);
    // Botón
    Button btnPonHora = findViewById(R.id.btnPonHora);
    btnPonHora.setOnClickListener(new View.OnClickListener() {
        @Override public void onClick() {
            ponHora();
        }
    });
}
```

- En cada momento hay una instrucción ejecutándose. **(NO HAY CONCURRENCIA)**
- Se dice que el programa es monotarea, monoproceto o monohebra (o single threading).

### PROGRAMA MULTITHILO



- En algún punto el programa se divide en varios procesos (threads) que se ejecutan (aparentemente) de manera simultánea.
- Es lo que se conoce como programa multiproceto, multitarea, multihebra (o multithreading)

# Programación concurrente.

## ¿Cómo expresar la concurrencia?

- Existen dos técnicas para producir actividades concurrentes en el computador.
  - **De forma automática**
    - El sistema operativo se encarga automáticamente (Por ejemplo, multiprogramación).
  - **De forma manual**
    - Mediante un **lenguaje concurrente de alto nivel**
      - Nuestro programa expresará acciones concurrentes (procesos e hilos), pero éstas no tienen por qué ejecutarse en paralelo.
      - Cada proceso concurrente se ejecuta en un procesador virtual
      - El compilador y el SO serán responsables de ejecutar nuestros procesos como consideren oportuno.

# Programación concurrente.

## ¿Cómo expresar la concurrencia?

### Lenguajes concurrentes de alto nivel

- Los lenguajes concurrentes de alto nivel incorporan características que permiten expresar la concurrencia directamente sin recurrir a servicios del S.O., bibliotecas, etc.
- Normalmente incluyen mecanismos de sincronización y comunicación entre procesos.
- Ejemplos: Ada, **Java**, etc

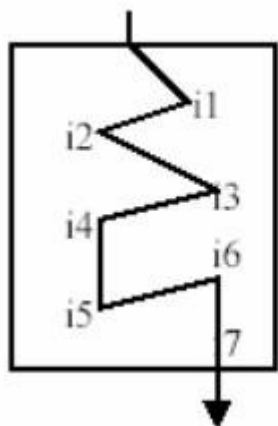
# **Características de los sistemas concurrentes**

- Orden de ejecución de las instrucciones
- Indeterminismo

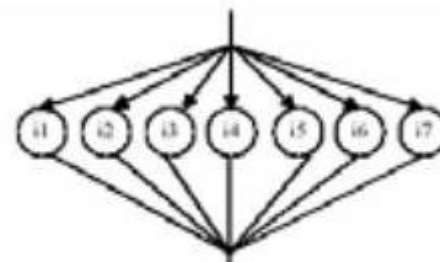
# Características de los sistemas concurrentes. Orden de ejecución de las instrucciones

- La **programación secuencial** define un orden total de las instrucciones
  - Ante un conjunto de datos de entrada el flujo de ejecución es siempre el mismo.
- Un **programa concurrente** define un orden parcial de ejecución
  - Ante un conjunto de datos de entrada no se puede saber cuál va a ser el flujo de ejecución.

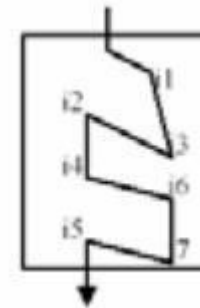
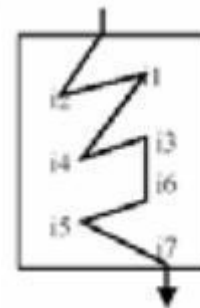
hilo de ejecución



programa



```
begin  
cobegin  
  i1; i2; i3; i4; i5; i6; i7;  
coned;  
end.
```



# Características de los sistemas concurrentes. Indeterminismo

- El orden parcial implica el no determinismo de los programas concurrentes
  - Es decir, puede producir diferentes resultados cuando se ejecuta repetidamente sobre el mismo conjunto de datos de entrada.
- Por ejemplo, ¿Qué valor tendrá la x al terminar el programa?

```
program Incognita;  
  var x: integer;  
  
  process P1;  
    var i: integer;  
  begin  
    for i:=1 to 5 do x:=x+1;  
  end;  
  
  process P2;  
    var j: integer;  
  begin  
    for j:=1 to 5 do x:=x+1;  
  end;
```

```
begin  
  x:=0;  
  cobegin  
    P1;  
    P2;  
  coend;  
end.
```

- El no determinismo es una propiedad inherente a la concurrencia.
- El indeterminismo origina que sea más difícil analizar y verificar un programa concurrente.
- Que existan varias posibilidades de salida NO significa necesariamente que un programa concurrente sea incorrecto.



# Beneficios de la programación concurrente (1)

- **Mejora el aprovechamiento de la CPU:** En sistemas monoprocesador permite optimizar los recursos. Se aprovechan las fases de E/S de una aplicación para procesamiento de otras.
- **Velocidad de ejecución**
  - **Ejemplos**
    - División de cálculos en procesos ejecutados en paralelo sobre plataformas multiprocesadoras: simulaciones, mercado electrónico...
    - Se puede ejecutar, por ejemplo, un lote mientras otro hilo lee el lote siguiente de un dispositivo.

# Beneficios de la programación concurrente (2)

- **Solución de problemas de naturaleza concurrente**
  - **Sistemas de control**
  - **Tecnologías web**
    - **Por ejemplo:** En un Servidor Web existe un proceso encargado de atender cada petición
  - **Aplicaciones basadas en interfaces de usuarios:** Mejora la interactividad de las aplicaciones: Se pueden separar las tareas de procesamiento de las tareas de atención de usuarios. Esta es precisamente una de las razones más convincentes para la multitarea.
    - **Por ejemplo:** En un programa de hoja de cálculo un hilo puede estar visualizando los menús y leer la entrada de usuario (atento a la interfaz gráfica) mientras que otro hilo ejecuta las órdenes y actualiza la hoja de cálculo.
  - Simulación

# Beneficios de la programación concurrente (3)

- **Facilita la programación**
  - Diversas tareas se pueden estructurar en procesos separados, lo que se consigue implementando cada tarea como clase independiente
- **Ventajas derivadas del procesamiento asíncrono**
  - **Ejemplo:** Las aplicaciones de procesamiento de textos guardan una copia de respaldo mientras se continua con la operación de escritura por el usuario sin interferir en la misma.

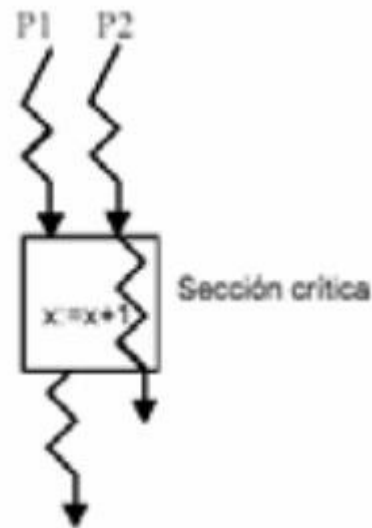
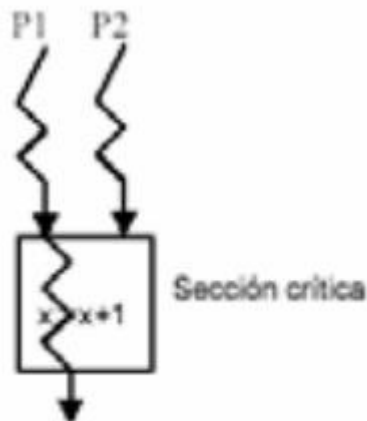
# Problemas inherentes a la programación concurrente

- Exclusión mutua
- Condición de sincronización
- Verificación

# Problemas inherentes a la programación concurrente.

## Exclusión mutua

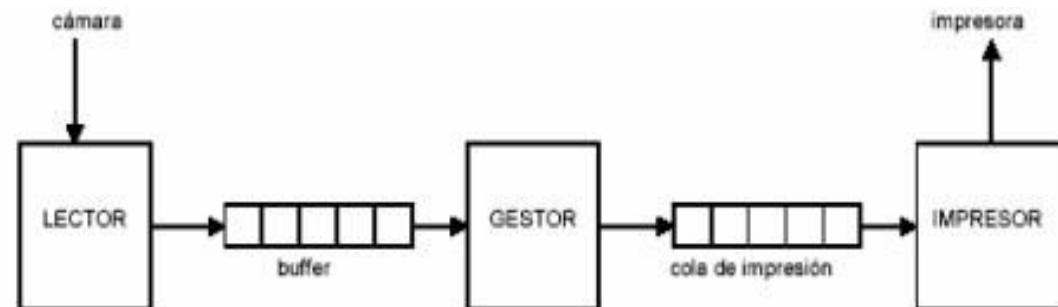
- La **exclusión mutua** consiste en impedir que dos o más procesos usen recursos compartidos.
- La **sección crítica** es la porción de código con variables compartidas y que debe ejecutarse en exclusión mutua.
  - Los lenguajes concurrentes deben proporcionar herramientas para resolver este tipo de problemas.



# Problemas inherentes a la programación concurrente.

## Condiciones de sincronización

- En un programa concurrente existen muchas posibilidades diferentes en cuanto al orden de ejecución.
- No obstante, pueden ser necesarias ciertas restricciones en el orden de ejecución.



```
process lector;  
begin  
  repeat  
    captura imagen;  
    almacena en buffer;  
  forever  
end;
```

```
process gestor;  
begin  
  repeat  
    coge imagen de buffer;  
    trata imagen;  
    almacena imagen en cola;  
  forever  
end;
```

```
process impresor;  
begin  
  repeat  
    coge imagen de cola;  
    imprime imagen;  
  forever  
end;
```

# Problemas inherentes a la programación concurrente.

## Verificación

- La corrección de programas concurrentes se puede determinar en base a una serie de propiedades.
- Existen **dos tipos de propiedades útiles en los programas concurrentes**:
  - **Propiedades de seguridad**
    - Son aquellas que aseguran que nada malo va a pasar durante la ejecución del programa.
    - Ejemplos: el algoritmo usado es el correcto, la exclusión mutua de regiones críticas, etc.
  - **Propiedades de viveza**
    - Son aquellas que aseguran que algo bueno pasará durante la ejecución del programa.
    - Ejemplos: si un proceso pide un recurso lo consigue en algún momento, los procesos no se bloquean mutuamente.
- Los programas concurrentes pueden no terminar nunca y al mismo tiempo ser correctos.
- Un programa concurrente **puede tener múltiples secuencias de ejecución**.
- **Cuando se dice que un programa concurrente es correcto, se entiende que se refiere a todas sus posibles secuencias de ejecución.**



# Aplicaciones de los programas concurrentes

- Aplicaciones clásicas
  - Programación de sistemas multicomputadores
  - Sistemas Operativos
  - Control y monitorización de sistemas físicos reales (por ejemplo, un sistema de control de temperatura y nivel)
- Aplicaciones actuales
  - Servicios WEB
  - Sistemas multimedia
  - Cálculo numérico
  - Procesamientos de entrada/salida...

# Conclusiones

- En este tema se ha tratado la diferencia entre **proceso y programa**.
- Se han estudiado aspectos relativos a la **concurrency**, se ha distinguido entre *concurrency software* y *parallelismo hardware* y se han tratado las diferentes *arquitecturas hardware* donde puede surgir la *concurrency*.
- Puesto que el concepto de concurrency va ligado indisolublemente al concepto de procesos, se han comentado brevemente aspectos sobre las dos unidades de ejecución básicas: **hilos y procesos**.
- Se ha hecho un recorrido sobre aspectos interesantes de la **programación concurrente**.

# Autoevaluación



- ¿Distingo entre programa y proceso?
- ¿Conozco bien el concepto de concurrencia?
- ¿Tengo claro el concepto de hilo y proceso, sus diferencias y la relación existente entre ambos?

# Actividades

- Realiza las actividades del tema correspondiente